

Partial-indistinguishability obfuscation using braids

Gorjan Alagic*, Stacey Jeffery†, and Stephen P. Jordan‡

Abstract

A circuit obfuscator is an algorithm that translates logic circuits into functionally-equivalent similarly-sized logic circuits that are hard to understand. While ad hoc obfuscators exist, theoretical progress has mainly been limited to no-go results. In this work, we propose a new notion of circuit obfuscation, which we call partial-indistinguishability. We then prove that, in contrast to previous definitions of obfuscation, partial-indistinguishability obfuscation can be achieved by a polynomial-time algorithm. Specifically, our algorithm re-compiles the given circuit using a gate that satisfies the relations of the braid group, and then reduces to a braid normal form. A variant of our obfuscation algorithm can also be applied to quantum circuits.

1 Introduction

Informally, an obfuscator is an algorithm that accepts a circuit as input, and outputs a hard-to-read but functionally equivalent circuit. (One can also discuss the related notion of obfuscating programs, but in this work we focus on obfuscating circuits.) Obfuscation methods used in practice so far have been essentially ad hoc [10, 35], and theoretical progress has primarily been in the form of no-go theorems for various strong notions of obfuscation. The ability to efficiently obfuscate certain circuits would have important applications in cryptography. For instance, sufficiently strong obfuscation of circuits of the form “encrypt with the private key” could turn a private-key encryption scheme into a public-key encryption scheme. As this example illustrates, one undesirable outcome is when the input circuit can be recovered completely from the obfuscated circuit. In this case, we say that the obfuscator *completely failed* on that circuit [6]. Unfortunately, every obfuscator will completely fail on some circuits; consider a circuit which is *learnable*, in the sense that a small number of its outputs can be used to efficiently compute a description of the circuit itself. On the other hand, there are trivial obfuscators which will erase at least some information from some circuits, e.g., by removing all instances of $X^{-1}X$ for some invertible gate X . These kinds of exceptions are part of the reason why giving good definitions of obfuscation and designing good obfuscators appear to be difficult.

In order to give a formal definition of obfuscation, one must decide on a reasonable definition of “hard-to-read.” The most stringent definition in the literature demands *black-box obfuscation*, i.e., that the output circuit is computationally no more useful than a black box that computes the same function. Barak et al. [7] gave an explicit family of circuits that are not learnable and yet cannot be black-box obfuscated. They also showed that there exists a private-key encryption scheme that

*Institute for Quantum Information, California Institute of Technology, Pasadena, CA.

†Institute for Quantum Computing, University of Waterloo, Waterloo, ON.

‡National Institute of Standards and Technology, Gaithersburg, MD. stephen.jordan@nist.gov

cannot be turned into a public-key cryptosystem by obfuscation. Their results do not preclude the possibility of black-box obfuscation for specific families of circuits, or of applying obfuscation to produce public-key systems from private ones in a non-generic fashion.

A more relaxed definition asks that different (but functionally-equivalent and similarly-sized) inputs to the obfuscator should lead to indistinguishable outputs. For any circuit C , let $|C|$ be the number of elementary gates, and let f_C be the Boolean function that C computes.

Definition 1. *A probabilistic algorithm \mathcal{O} is an indistinguishability obfuscator for the collection \mathcal{C} of circuits if the following three conditions hold:*

1. *(functionality) for every $C \in \mathcal{C}$, $f_{\mathcal{O}(C)} = f_C$;*
2. *(polynomial slowdown) there is a polynomial p such that $|\mathcal{O}(C)| \leq p(|C|)$ for every $C \in \mathcal{C}$;*
3. *(indistinguishability obfuscation) For sufficiently large input lengths, and for any $C_1, C_2 \in \mathcal{C}$ such that $f_{C_1} = f_{C_2}$ and $|C_1| = |C_2|$, the two distributions $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ are indistinguishable.*

In the above definition, the map \mathcal{O} may be a probabilistic map, in which case one must choose a notion of indistinguishability for probability distributions. Goldwasser and Rothblum [19] consider three such notions: perfect (exact equality), statistical (total variation distance bounded by a constant), and computational (no probabilistic polynomial-time Turing Machine can distinguish samples with better than negligible probability.) They show that the existence of an efficient statistical indistinguishability obfuscator would result in a collapse of the polynomial hierarchy to the second level. This result also applies if the condition $|C_1| = |C_2|$ in property (3) of definition 1 is relaxed to $|C_1| = k|C_2|$ for any fixed constant k [19].

We remark that an indistinguishability obfuscator does not immediately provide a generic method for turning private keys into public keys. To see this, consider a family of encryption circuits $\{E_k\}$ corresponding to private keys k , and suppose it is easy to recover k from the circuit diagram of E_k . Consider the algorithm that computes the entire function table of the input circuit C , and outputs E_k if the functions implemented by C and E_k are equal. Since each E_k has different functionality, it's not hard to see that this algorithm is an (inefficient) indistinguishability obfuscator for the family of all circuits which are equivalent to (and of similar size as) the encryption circuits. The “obfuscation” of E_k is just itself, and clearly cannot serve as a public key. Note that if this functionality can be achieved via an efficient algorithm \mathcal{O} , then no other obfuscator \mathcal{O}' can successfully hide the keys, since given $\mathcal{O}'(E_k)$, one can simply compute $\mathcal{O}(\mathcal{O}'(E_k)) = E_k$.

Another natural choice of property (3) in Definition 1 is *best-possible obfuscation*; in that case, we ask that the obfuscated circuit reveals no more information than any other circuit that computes the same function. Goldwasser and Rothblum [19] showed that for efficient obfuscators, indistinguishability obfuscation is equivalent to best-possible obfuscation. They then proved that, in the limited computational model of polynomial-sized ordered binary decision diagrams (or POBDDs), perfect indistinguishability obfuscation is possible but black-box obfuscation is not. The key fact is that POBDDs have an efficiently computable normal form [9]. The obfuscator simply computes that normal form, perfectly satisfying property (3) in Definition 1.

For general Boolean circuits, an efficiently computable normal form is too much to ask for, as deciding circuit equivalence is coNP-hard. Our approach is to instead pursue a notion of “partial”

normal form. In partial-indistinguishability obfuscation, we relax condition (3) so that it need only hold for C_1 and C_2 that are related by some fixed, finite set of relations on the underlying gate set¹

Definition 2. Let G be a set of gates and Γ a set of relations satisfied by the elements of G . A probabilistic algorithm \mathcal{O} is a (G, Γ) -indistinguishability obfuscator for the collection \mathcal{C} of circuits over G if the following three conditions hold:

1. (functionality) for every $C \in \mathcal{C}$, $f_C = f_{\mathcal{O}(C)}$;
2. (polynomial slowdown) there is a polynomial p such that for every $C \in \mathcal{C}$, $n_{\mathcal{O}(C)} \leq p(n_C)$ and $|\mathcal{O}(C)| \leq p(|C|)$;
3. ((G, Γ) -indistinguishability obfuscation) For any $C_1, C_2 \in \mathcal{C}$ that differ only by a sequence of applications of the relations in Γ , the two distributions $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ are indistinguishable.

The power of the obfuscation is then determined by the power of the relations Γ . If Γ is a complete set of relations, generating all circuit equivalences over G , then a (G, Γ) -indistinguishability obfuscator is essentially a perfect indistinguishability obfuscator, as defined in [19]. (Complete sets of relations for $\{\text{Toffoli}\}$ and $\{\text{AND}, \text{OR}, \text{NOT}\}$ are given in [24, 23].) In the other extreme, if Γ is the empty set then a (G, Γ) -obfuscator does not hide a circuit at all; any two circuits are mapped to distinguishable distributions. With different sets of relations, one can interpolate between these extremes. The intermediate obfuscators form a partially ordered set, where a (G, Γ') -indistinguishability obfuscator is strictly stronger than a (G, Γ) -indistinguishability obfuscator if Γ' is a strict superset of Γ .

In this paper, we propose an efficient obfuscator for reversible circuits, where the gate set comes from computationally universal representations of the braid group, and the relations are the braid relations. Specifically, in what follows, **B** accepts a circuit input and outputs the corresponding braid, **C** accepts a braid input and outputs the corresponding circuit, and **N** accepts a braid input and computes the normal form braid. The underlying representation is the quantum double of A_5 , as discussed in section 4. Our obfuscation algorithm is simply

Algorithm 1.

1. input: a circuit C on n dits
2. output: The circuit $\mathbf{C}(\mathbf{N}(\mathbf{B}(C)))$.

Our scheme is thus similar in spirit to previously-proposed obfuscation schemes based on applying local circuit identities [35], but the uniqueness of normal forms adds a qualitatively new feature. In our algorithm, each gate is simulated with a constant-size braid, and each braid crossing is simulated with a constant-size circuit; these algorithms are described in Section 4. Computing the normal form takes time $O(l^2 m \log m)$ for m -strand braids of length l ; the relevant background in braid groups and their normal forms is given in Section 2. Putting these ingredients together, we see that the overall time complexity of Algorithm 1 is $O(|C|^2 n \log n)$.

In Section 5, we describe how a variant of our scheme can also be used to obfuscate quantum circuits. The key (well-known) fact is that there are representations of the braid group which are universal for quantum computation. The quantum obfuscator has time complexity $O(|C|^2 n \cdot \text{polylog}(n, 1/\epsilon))$ if one wishes to achieve functional equivalence to precision ϵ .

¹Our construction for satisfying this definition uses *reversible* gates. The definition of functional equivalence becomes more technical that context, as discussed in section 3.

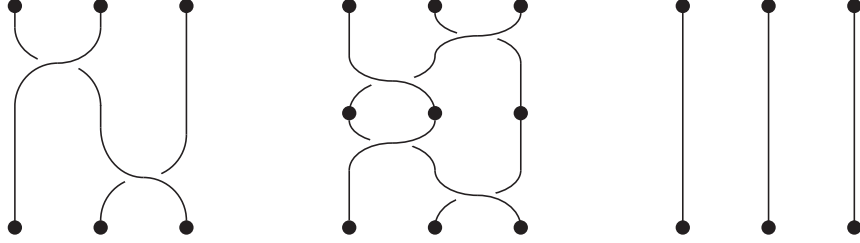


Figure 1: The generator σ_i represents the (clockwise) crossing of strands i and $i + 1$ connecting a bottom row of “pegs” to a top row. Multiplication of group elements corresponds to composition of braids. As an example, we show the 3-strand braid $\sigma_1^{-1}\sigma_2$ (left), and the same braid composed with its inverse $\sigma_2^{-1}\sigma_1$ (middle), which is equivalent to the identity element of B_3 (right).

2 Braid groups

The braid group B_n is the infinite discrete group with generators $\sigma_1, \dots, \sigma_{n-1}$ and relations

$$\begin{aligned} \sigma_i \sigma_j &= \sigma_j \sigma_i & \forall |i - j| \geq 2 \\ \sigma_i \sigma_{i+1} \sigma_i &= \sigma_{i+1} \sigma_i \sigma_{i+1} & \forall i. \end{aligned} \tag{1}$$

The group B_n is thus the set of all words in the alphabet $\{\sigma_1, \dots, \sigma_{n-1}, \sigma_1^{-1}, \dots, \sigma_{n-1}^{-1}\}$, up to equivalence determined by the above relations. In 1925 Artin proved that the abstract group defined above precisely captures the topological equivalence of braided strings [5], as illustrated in Fig. 1. A charming exposition of this subject can be found in [27].

In the word problem on B_n , we are given words w and z , and our goal is to determine if they are equal as elements of B_n . One solution is to put both w and z into a *normal form*, and then check if they are equal as words. For our purposes, it is enough to describe the normal form and specify the complexity of the algorithm for computing it. The details of the algorithm, along with a thorough and accessible presentation of the relevant facts about braids, can be found in [13].

We first observe that the word problem is easily decidable if we restrict our attention to an important subset of B_n . Note that the presentation (1) can also be viewed as a presentation of a monoid, which we denote by B_n^+ . The elements of B_n^+ are called *positive braids*, and are words in the generators σ_i only (no inverses), up to equivalence determined by the relations in (1). Since all the relations of B_n preserve word length, and there are only finitely many words of any given length, we can decide the word problem (albeit very inefficiently) simply by trying all possible combinations of the relations.

Building upon this, one can give an (inefficient) algorithm for the word problem on B_n itself [20]. First, given two elements a, b of B_n^+ , we write $a \preceq b$ if there exists $z \in B_n^+$ such that $b = az$; in this case we say that a is a *left divisor* of b . Similarly, we write $a \succeq b$ if there exists $y \in B_n^+$ such that $b = ya$; in this case we say that a is a *right divisor*² of b . The center of B_n is the cyclic group generated by Δ_n^2 , where

$$\Delta_n := \Delta_{n-1} \sigma_{n-1} \sigma_{n-2} \cdots \sigma_1 \in B_n^+$$

²The terminology is not accidental; it turns out that we can also define l.c.m.s and g.c.d.s in B_n^+ , and that B_n is the group of fractions of B_n^+ . These facts are some of the achievements of Garside theory [18].

(see p.30 of [20] for a simple proof). Geometrically, Δ_n implements a twist by π in the z -plane as the strands move from $z = 0$ to $z = 1$. One can show that $\sigma_i \preceq \Delta_n$ for all i , i.e. there exists $x_i \in B_n^+$ such that $\sigma_i^{-1} = x_i \Delta_n^{-1}$. Given a word w in the σ_i and their inverses, we first replace the leftmost instance of an inverse generator (say it is σ_i^{-1}) with $x_i \Delta_n^{-1}$. We then insert $\Delta_n^{-1} \Delta_n$ in front of x_i , and observe that conjugating a positive braid x by Δ_n results in another positive braid (specifically, the rotation of x by π in the z -plane). In this way, we can push Δ_n^{-1} all the way to the left. We repeat this process for each inverse generator appearing in the word, resulting in a word of the form $\Delta_n^p b$ where $p \in \mathbb{Z}$ and $b \in B_n^+$. Since we can solve the word problem in B_n^+ , we can factor out the maximal power of Δ_n appearing as a left divisor of b . We thus have that, as elements of the braid group, $w = \Delta_n^{p'} b'$ with Δ_n not a left divisor of b' and p' unique. This solves the word problem in B_n .

We can make the above algorithm efficient by finding an efficiently computable normal form for a positive braid word b that does not have Δ_n as a left divisor. Recall that the symmetric group S_n has a remarkably similar presentation to B_n . Indeed, starting with (1), letting $\sigma_i = (i \ i+1)$ and adding the relations $\sigma_i^2 = 1$ for all i results in the standard presentation of S_n . In other words, there is a surjective homomorphism $\pi : B_n \rightarrow S_n$ with $\sigma_i \mapsto (i \ i+1)$. In terms of the geometric interpretation, a braid is mapped to the permutation on $[n]$ defined by the connections between the top and bottom “pegs,” as in Figure 1. For each $\sigma \in S_n$, there is a unique preimage of σ that can be drawn so that any given pair of strands cross only in the positive direction, and at most once. We call such braids *simple braids*, and they form a subset of B_n^+ of size $n!$.

Definition 3. p.4 of [13].

1. A sequence of simple braids (s_1, \dots, s_p) is said to be normal if, for each j , every σ_i that is a left divisor of s_{j+1} is a right divisor of s_j .
2. A sequence of permutations (f_1, \dots, f_p) is said to be normal if, for each j , $f_{j+1}^{-1}(i) > f_{j+1}^{-1}(i+1)$ implies $f_j(i) > f_j(i+1)$.

A sequence of simple braids (s_1, \dots, s_p) is normal if and only if the sequence of permutations $(\pi(s_1), \dots, \pi(s_p))$ is normal. Given a permutation $f \in S_n$, let \hat{f} denote the simple braid of B_n satisfying $\pi(\hat{f}) = f$.

Theorem 1. p.4 of [13] and Ch.9 of [14].

1. Every braid z in B_n admits a unique decomposition of the form $\Delta_n^m s_1 \dots s_p$ with $m \in \mathbb{Z}$ and (s_1, \dots, s_p) a normal sequence of simple braids satisfying $s_1 \neq \Delta_n$ and $s_p \neq 1$.
2. Every braid z in B_n admits a unique decomposition of the form $\Delta_n^m \hat{f}_1 \dots \hat{f}_p$ with $m \in \mathbb{Z}$ and (f_1, \dots, f_p) a normal sequence of permutations satisfying $f_1 \neq \pi(\Delta_n)$ and $f_p \neq 1$.

The most efficient algorithms for computing the normal form of a word w in the generators of B_n have complexity $O(|w|^2 n \log n)$ [13].

3 Reversible Circuits

The partial-indistinguishability obfuscator given in section 4 uses a gate R satisfying the relations of the braid group. Because group elements are invertible, R must be a reversible gate, that is,

it must bijectively map its possible inputs to its possible outputs. For example, a NOT gate is reversible, but an AND gate is not. In general, a reversible circuit need not act on bits, but can act on d -state dits. For bijectivity, the number of output dits must equal the number of input dits. A circuit composed entirely of reversible gates is called a reversible circuit. For more background on reversible computation see [8, 16, 32].

Because reversible circuits cannot erase any information, they operate using ancillary dits (“ancillas”) to store unerasable data left over from intermediate steps in the computation. A reversible circuit evaluating a function $f : \{0, \dots, d-1\}^n \rightarrow \{0, \dots, d-1\}^m$ thus operates on $r \geq \max(n, m)$ dits, where $r - n$ of the input dits are work dits to be initialized to some fixed value independent of the problem instance, and $r - m$ of the output dits contain unerasable leftover data, to be ignored. Efficient procedures are known for compiling arbitrary logic circuits into reversible form [8, 16].

In definition 1 of perfect indistinguishability obfuscation, the notion of functional equivalence is used twice. First, the original circuit C must be functionally equivalent to the obfuscated circuit $\mathcal{O}(C)$. Second, if C_1 and C_2 are functionally equivalent then $\mathcal{O}(C_1)$ must be indistinguishable from $\mathcal{O}(C_2)$. In partial-indistinguishability obfuscation, the second usage of functional equivalence is superseded by the set of relations Γ .

In adapting definitions 1 and 2 to reversible circuits, one is faced with two natural choices for the notion of functional equivalence. One may either demand that the original and obfuscated circuits implement the same function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, ignoring the ancilla dits (*weak equivalence*), or demand that they implement the same transformation on the entire set of r dits, including the ancillas (*strong equivalence*).

The construction given in this section satisfies partial-indistinguishability obfuscation so that the obfuscated circuit is strongly equivalent to the original circuit. Strong equivalence implies weak equivalence, so our construction proves that both possible definitions of partial-indistinguishability are polynomial-time achievable when Γ is the set of relations of the braid group.

One is left with a natural question: is perfect indistinguishability obfuscation of reversible circuits possible if we only demand that $\mathcal{O}(C_1) = \mathcal{O}(C_2)$ when C_1 is strongly equivalent to C_2 ? In the case of ordinary irreversible circuits, we argued that perfect polynomial-time deterministic indistinguishability obfuscation is impossible (assuming $P \neq NP$) because circuit equivalence is coNP-complete. As shown in [26], strong equivalence of reversible circuits remains coNP-complete for standard reversible gate sets. Thus, assuming $P \neq NP$, deterministic indistinguishability obfuscation of reversible circuits cannot be achieved in polynomial time even if one only demands that strongly equivalent circuits have indistinguishable obfuscations.

4 Classical Computation with braids

In this section, we present a reversible gate R on pairs of 60-state dits that can perform universal computation and obeys the relations of the braid group. The universality construction for this gate comes from the quantum computation literature [28, 33, 30], but we present it here in purely classical language to make it accessible to a broader audience.

Suppose we arrange n dits on a line, and allow R to act only on neighboring dits. Further, we do not allow R to be applied “upside-down”. Then, there are $n - 1$ choices for how to apply R . We label these R_1, R_2, \dots, R_{n-1} , as illustrated in Figure 2. Each of R_1, \dots, R_{n-1} corresponds to a $d^n \times d^n$ permutation matrix. Specifically, R_j is obtained by taking the tensor product of R with identity matrices according to $R_j = \mathbb{1}_{d \times d}^{\otimes(j-1)} \otimes R \otimes \mathbb{1}_{d \times d}^{\otimes(n-j-1)}$.

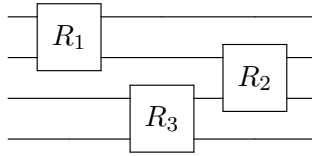


Figure 2: An example of a reversible circuit constructed from a single gate R . As a product of matrices, we write this $R_2R_3R_1$, in keeping with the convention [32] that circuit diagrams are to be read left-to-right, whereas the matrix product acts right-to-left. Note that in subsequent circuit diagrams we drop the subscripts from the R gates as these can be read off from the “wires” the gates act on.

R_1, \dots, R_{n-1} generate a subgroup of S_{d^n} . Among others, these generators obey the relations

$$R_i R_j = R_j R_i \quad \forall |i - j| \geq 2. \quad (2)$$

If R satisfies

$$R_1 R_2 R_1 = R_2 R_1 R_2 \quad (3)$$

then

$$R_i R_{i+1} R_i = R_{i+1} R_i R_{i+1} \quad \forall i \quad (4)$$

and in this case the gates R_1, \dots, R_{n-1} satisfy all the relations of the braid group B_n . In other words, the map defined by $\sigma_i \mapsto R_i$ and $\sigma_i^{-1} \mapsto R_i^{-1}$ is a homomorphism from B_n to S_{d^n} , *i.e.* a representation of the braid group. Note that this representation is never faithful as B_n is infinite.

The condition 3 is known as the Yang-Baxter equation³. Finding all the matrices satisfying the Yang-Baxter equation at a given dimension has only been achieved at $d = 2$ [22]. However, certain systematic constructions coming from mathematical physics can produce infinite families of solutions. In particular, let G be any finite group, and let R be the permutation on the set $G \times G$ defined by

$$R(a, b) = (b, b^{-1}ab). \quad (5)$$

By direct calculation one sees that any such an R satisfies the Yang-Baxter equation. (In physics language, R comes from the braiding statistics of the magnetic fluxes in the quantum double of G .)

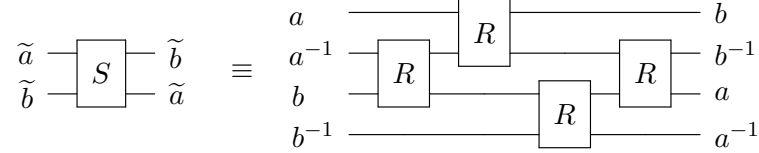
In 1997, Kitaev discovered that choosing G to be the symmetric group S_5 yields an R gate sufficient to perform universal reversible computation [28]. Ogburn and Preskill subsequently showed that the alternating group A_5 , which is half as large as S_5 , is already sufficient. The universality construction for A_5 was subsequently presented in greater detail and generalized to all non-solvable groups by Mochon [30]. In the remainder of this section we give a self-contained exposition of the universality construction from [30], shorn of physics language.

To obtain a representation of the braid group, we must strictly enforce the requirement that application of R to neighboring dits on a line is the only allowed operation. In particular, we are not given as elementary operations the ability to apply R upside-down, or to non-neighboring dits,

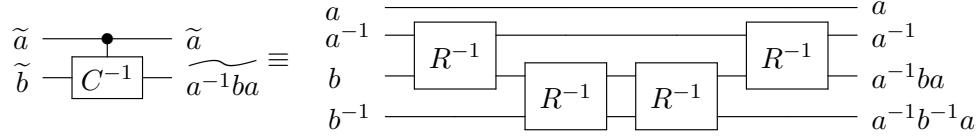
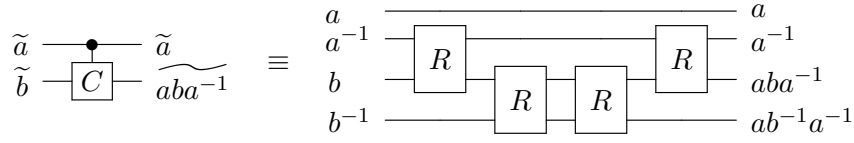
³Actually, two slightly different equations go by the name Yang-Baxter in the literature. Careful sources distinguish these as the algebraic Yang-Baxter equation and the braided Yang-Baxter relation (which is sometimes called the quantum Yang-Baxter equation). Equation 3 is the latter. Furthermore, some sources treat a more complicated version of the Yang-Baxter equation in which R depends on a continuous parameter. In such works equation 3 is often referred to as the constant Yang-Baxter equation.

or to move dits around. Thus, to prove computational universality, it is helpful to first construct a SWAP gate from R gates, which exchanges neighboring dits. As is well-known, the $n - 1$ swaps of nearest neighbors on a line generate the full group S_n of permutations, and thus a SWAP gate enables application of R to any pair of dits.

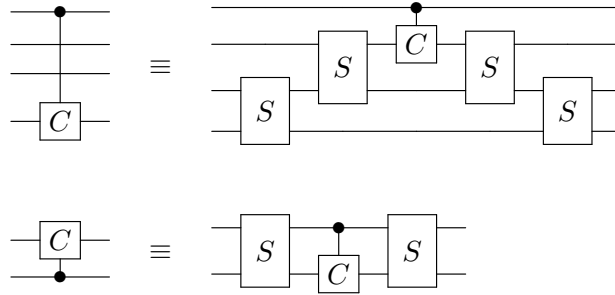
For R gates of the form (5), two pairs of inverse group elements in the order a, a^{-1}, b, b^{-1} can be swapped by applying the product $R_2 R_3 R_1 R_2$. Thus, in the construction of [33, 30], elements of A_5 are always paired with their inverses. This can be regarded as a form of encoding; $|A_5| = 60$, so each 60-state dit is encoded by a corresponding pair of elements of A_5 . We introduce the notation $\tilde{g} \equiv (g, g^{-1})$ for this encoding, and similarly, abbreviate the encoded swap operation as follows.



Similarly, the sequence $R_2 R_3 R_3 R_2$ performs the transformation $(\tilde{a}, \tilde{b}) \mapsto (\tilde{a}, \widetilde{aba^{-1}})$ on a pair of encoded dits. We abbreviate this in circuit diagrams as follows.



This notation can easily be extended to provide a shorthand for the sequence of gates needed to implement a C gate between non-neighboring pairs of bits, as illustrated by the following examples.



Next, consider the following product of elements of A_5 (which should be read right-to-left).

$$f(g_1, g_2) = (521)g_1(14352)g_2(124)g_1^{-1}(15342)g_2^{-1}(521) \quad (6)$$

One sees that

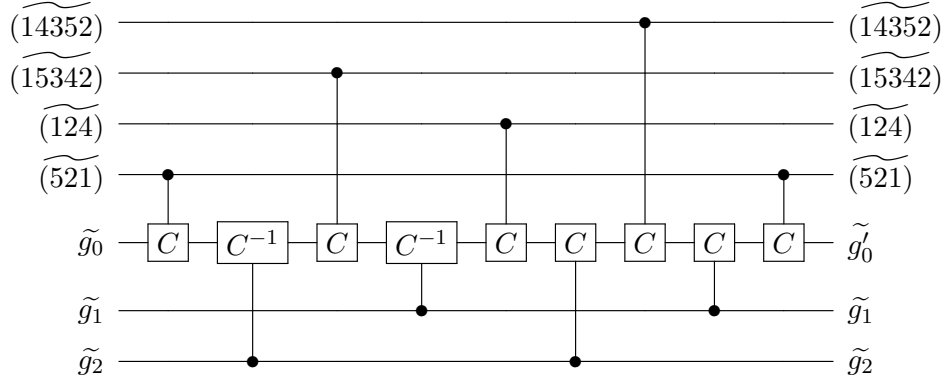
$$\begin{aligned} f((345), (345)) &= \mathbb{1} \\ f((345), (435)) &= \mathbb{1} \\ f((435), (345)) &= \mathbb{1} \\ f((435), (435)) &= (12)(34) \end{aligned}$$

where $\mathbb{1}$ denotes the identity permutation. Furthermore, conjugating (345) by (12)(34) yields (435), and conversely, conjugating (435) by (12)(34) yields (345). Thus, we may think of (345) as an encoded zero and (435) as an encoded one, and we see that

$$f(g_1, g_2)g_0f(g_1, g_2)^{-1} \quad (7)$$

toggles g_0 between one and zero if g_1 and g_2 are both encoded ones and leaves g_0 unchanged otherwise. Such a doubly-controlled toggling operation is known as a Toffoli gate, which is well-known to be a computationally universal reversible gate [16].

As a circuit diagram, this construction can be expressed as follows.



Here, if g_0, g_1, g_2 encode bits b_0, b_1, b_2 then g_0' encodes $b_0 \oplus b_1 \wedge b_2$. The four ancillary dits (14352) , (15342) , (124) , and (521) , are used to “catalytically” facilitate the construction of a Toffoli gate, and thus computations built from arbitrarily many Toffoli gates can be performed with only one copy of these four dits.

Unpacking the various shorthand notations, one sees that the above circuit represents the following braid of 132 crossings on 14 strands.

$$\begin{aligned} T = & \begin{array}{cccc} \sigma_8\sigma_9\sigma_9\sigma_8 & \sigma_{10}\sigma_{11}\sigma_9\sigma_{10} & \sigma_{10}\sigma_{11}\sigma_{11}\sigma_{10} & \sigma_{10}\sigma_{11}\sigma_9\sigma_{10} \\ \sigma_2\sigma_3\sigma_1\sigma_2 & \sigma_4\sigma_5\sigma_3\sigma_4 & \sigma_6\sigma_7\sigma_5\sigma_6 & \sigma_8\sigma_9\sigma_9\sigma_8 \\ \sigma_6\sigma_7\sigma_5\sigma_6 & \sigma_4\sigma_5\sigma_3\sigma_4 & \sigma_2\sigma_3\sigma_1\sigma_2 & \sigma_{12}\sigma_{13}\sigma_{11}\sigma_{12} \\ \sigma_{10}\sigma_{11}\sigma_9\sigma_{10} & \sigma_{10}\sigma_{11}\sigma_{11}\sigma_{10} & \sigma_{10}\sigma_{11}\sigma_9\sigma_{10} & \sigma_{12}\sigma_{13}\sigma_{11}\sigma_{12} \\ \sigma_6\sigma_7\sigma_5\sigma_6 & \sigma_8\sigma_9\sigma_9\sigma_8 & \sigma_6\sigma_7\sigma_5\sigma_6 & \sigma_{10}\sigma_{11}\sigma_9\sigma_{10} \\ \sigma_{10}^{-1}\sigma_{11}^{-1}\sigma_{11}^{-1}\sigma_{10}^{-1} & \sigma_{10}\sigma_{11}\sigma_9\sigma_{10} & \sigma_4\sigma_5\sigma_3\sigma_4 & \sigma_6\sigma_7\sigma_5\sigma_6 \\ \sigma_8\sigma_9\sigma_9\sigma_8 & \sigma_6\sigma_7\sigma_5\sigma_6 & \sigma_4\sigma_5\sigma_3\sigma_4 & \sigma_{12}\sigma_{13}\sigma_{11}\sigma_{12} \\ \sigma_{10}\sigma_{11}\sigma_9\sigma_{10} & \sigma_{10}^{-1}\sigma_{11}^{-1}\sigma_{11}^{-1}\sigma_{10}^{-1} & \sigma_{10}\sigma_{11}\sigma_9\sigma_{10} & \sigma_{12}\sigma_{13}\sigma_{11}\sigma_{12} \\ \sigma_8\sigma_9\sigma_9\sigma_8 & & & \end{array} \end{aligned}$$

Note that we take the convention that this should be read backwards compared to the way one reads English text. This is in keeping with the conventional notation for the composition of functions

and our right-to-left multiplication of R matrices. We have used whitespace to divide crossings into groups of four as these correspond to elementary S and R gates.

Given this construction of the Toffoli gate by braid crossings, it is a simple matter to “compile” any given logic circuit into a corresponding braid. There are 3600 pairs of elements of A_5 . Thus, encoding a single bit into a pair of A_5 elements appears somewhat wasteful. We do not know the smallest dimension of a permutation matrix satisfying the Yang-Baxter equation that acts as a universal reversible gate, but we have proven by exhaustive computer search that it is at least 25×25 .

5 Quantum circuits

5.1 Obfuscating quantum circuits with braids

While the state of knowledge about classical obfuscation is limited, essentially nothing is known about the quantum case. Here we briefly discuss how to construct an obfuscator for quantum circuits, analogously to the classical obfuscator defined by Algorithm 1.

In section 4, we discussed classical universality of circuits encoded as braids. It turns out that an analogous theory can be developed for quantum circuits, and is well-understood. The family of so-called Fibonacci representations of the braid groups have dense image in the unitary group, and there are efficient classical algorithms for translating any quantum circuit into a braid (and vice-versa) in a way that preserves unitary functionality [17]. Approachable descriptions of the Fibonacci representation are given in [34, 36]. In [34], what we call the “Fibonacci representation” here, is called the “ $\star\star$ ” irreducible sub-representation. This is a family of representations $\rho_{\text{Fib}}^{(n)} : B_n \rightarrow U(F_{n-4})$, where F_k is the k -th Fibonacci number. For our application, the essential properties of the Fibonacci representation are *locality* and *local density*. These two properties mean that, under a certain qubit encoding, braid generators correspond to local unitaries, and local unitaries correspond to short braid words. Standard arguments from quantum computation tell us that we can achieve the latter to precision ϵ with $O(\log^{2.71}(1/\epsilon))$ braid generators by means of the Solovay-Kitaev algorithm [12].

A natural basis for the space of $\rho_{\text{Fib}}^{(n)}$ can be identified with strings of length n from the alphabet $\{\star, p\}$, which begin with \star , end with p , and do not contain “ $\star\star$ ” as a substring⁴. Following [2]⁵, for n a multiple of four, we identify a particular subspace V_n of $\rho_{\text{Fib}}^{(n)}$ by discarding some basis elements, as follows. Partition a string s into substrings of length four. If each of these substrings is equal to either $\star p \star p$ (this will encode a 0) or $\star p p p$ (this will encode a 1), then the basis element corresponding to s is in V_n ; otherwise, it is not. Note that $\dim V_n = 2^{n/4}$. The following theorem follows from [2, 12].

Theorem 2. *There is a classical algorithm which, given an $(n/4)$ -qubit quantum circuit C and $\epsilon > 0$, outputs a braid $b \in B_n$ of length $O(|C| \log^{2.71}(1/\epsilon))$ satisfying*

$$\left\| C - \rho_{\text{Fib}}^{(n)}(b) \Big|_{V_n} \right\| \leq \epsilon ;$$

⁴In [34] the $\star\star$ subrepresentation of B_n acts on strings of length $n + 1$ that begin and end with \star . One can leave the initial and/or final \star implicit as these are left unchanged by all braiding operations. We omit the final \star leaving us strings of length n that begin with \star and end with p .

⁵Reference [2] describes the basis vectors in terms of “paths”. The correspondence between the path notation and the $p\star$ notation is given in appendix C of [34].

this algorithm has complexity $O(|b|)$.

For the opposite direction, we can identify a subspace $W_n \subset (\mathbb{C}_2)^{\otimes n}$ by discarding all bitstrings except those that start with 0, end with 1 and do not have “00” as a substring. Then $\dim W_n = \dim \rho_{\text{Fib}}^{(n)}$ and we have the following.

Theorem 3. *There is a classical algorithm which, given $b \in B_n$ and $\epsilon > 0$, outputs a quantum circuit C on n qubits of length $O(|b| \log^{2.71}(1/\epsilon))$ such that*

$$\left\| C|_{W_n} - \rho_{\text{Fib}}^{(n)}(b) \right\| \leq \epsilon ;$$

this algorithm has complexity $O(|C|)$.

The two algorithms in the above theorems are described explicitly in [2]. With these algorithms in hand, we can apply Algorithm 1 directly to quantum circuits. For an input circuit C on n qubits, the running time of the algorithm is $O(|C|^2 n \cdot \text{polylog}(n, 1/\epsilon))$. The length of the output cannot be longer than the running time. We are not aware of a better upper bound for the length of the output.

Note that reduction of quantum circuits to a normal form using a complete set of gate relations should not be possible in polynomial time, because this would yield a polynomial-time algorithm for deciding whether a quantum circuit is equivalent to the identity, which is a coQMA-complete problem [25].

In light of theorem 2, $\{\rho_{\text{Fib}}(\sigma_1), \dots, \rho_{\text{Fib}}(\sigma_{n-1})\}$ may be regarded as a universal set of elementary quantum gates. A word in the Artin generators of B_n then corresponds to a circuit in this gate set. The “gates” $\{\rho_{\text{Fib}}(\sigma_1), \dots, \rho_{\text{Fib}}(\sigma_{n-1})\}$ differ from conventional quantum gates in that they do not possess locality defined in terms of a strict tensor product structure. Nevertheless, the algorithm for computing the braid normal form satisfies the definition of partial-indistinguishability obfuscation for circuits composed from the gates $\{\rho_{\text{Fib}}(\sigma_1), \dots, \rho_{\text{Fib}}(\sigma_{n-1})\}$.

5.2 Testing claimed quantum computers with a quantum obfuscator

It is natural to consider quantum analogues of the applications of obfuscation from classical computer science. We suggest a potential application of quantum circuit obfuscation that does not fit this mold: testing claimed quantum computers. Suppose Bob claims to have access to a universal quantum computer with some fixed finite number of qubits. Alice has access to a classical computer only, as well as a classical communication channel with Bob. Can Alice determine if Bob is telling the truth? Barring tremendous advances in complexity theory, a provably correct test is unlikely⁶; can we still design a test in which we have a high degree of confidence? Given the extensive work on classical algorithms for factoring, a reasonable idea is to simply ask Bob to factor a sufficiently large RSA number. However, Shor’s algorithm only begins to outperform the best classical algorithms when thousands of logical qubits can be employed. A much smaller universal quantum computer (e.g., a few dozen qubits) is likely to be a far simpler engineering challenge and could still be quite useful, e.g., for simulating certain quantum systems. A test that works in this case would thus

⁶Notice that even a proof that $\text{BQP} \neq \text{BPP}$ would be insufficient; one would have to find specific problems and instance sizes where some quantum strategy provably beats every classical one. We are thus left with a situation analogous to the practical security guarantees of modern cryptographic systems, which tell us how many bit operations it would take to crack a given instance using the fastest known algorithms.

be very valuable. We now outline a new proposal for such a test. Simply put, we propose asking questions that are classically easy to answer, but posing them in an obfuscated manner. In this test, Alice would repeatedly generate quantum circuits and ask Bob to run them. At least some of the circuits would in fact be quantumly-obfuscated classical reversible circuits, allowing Alice to easily check the answers.

We have considerable freedom when designing such a test. How to choose these parameters in a way that makes the test difficult to fool with a classical computer is an open question. For purposes of illustration, we give one example. Let \mathcal{O} be the obfuscation algorithm for quantum circuits described above.

Algorithm 2.

1. Select a random bitstring s of length k .
2. Let C be the $(k + 1)$ -bit circuit that, on all-zero input, initializes wires 2 through $k + 1$ to s and then computes the parity of s into the first wire.
3. Compute $\mathcal{O}(C)$, and let n be the number of qubits needed to run $\mathcal{O}(C)$.
4. Ask Bob to run D on the all zeros string and return the first bit of output.

Clearly, k must be chosen so that n is smaller than the number of logical qubits Bob claims to control. To fool Alice, a purely classical Bob must determine the parity of s . The dictionary attack (*i.e.* Bob repeatedly guesses at k , obfuscates the corresponding circuit, and compares the result to the circuit given by Alice) is of no use provided k is reasonably large, e.g., 80 bits, which can be encoded using a braid of 115 strands using the Zeckendorf encoding described in [34].

We now show that there can be no efficient general-purpose algorithm for breaking our test by detecting whether a given quantum circuit is in fact (almost) classical, and if so, simulating it.

Definition 4. *Let c be a bit string specifying a quantum circuit via a standard universal set Q of quantum gates, and let U_c be the corresponding unitary operator. Fix some constants $r, d, a \in \mathbb{N}$, and fix a set R of reversible gates. The problem $\text{CLASS}(r, d, a, Q, R)$ is to find a reversible circuit of at most $r|c|^d$ gates from R such that the corresponding permutation matrix P satisfies $\|U_c - P\| \leq 2^{-a|c|}$.*

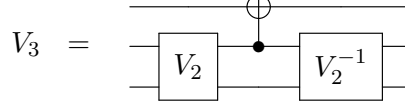
Note that $\text{CLASS}(r, d, a, Q, R)$ is not a decision problem. Thus, to formulate the question of whether this problem can be efficiently solved, we must ask not whether $\text{CLASS}(r, d, a, Q, R)$ is contained in P but whether it is contained in FP. We now provide some formal evidence that this is not the case. Note that the following theorems continue to hold if we change the classicality condition in 4 to $\|U_c - P\| \leq |c|^{-a}$.

Theorem 4. *For any fixed $r, d, a \in \mathbb{N}$, any universal reversible gate set R , and any universal quantum gate set Q , if $\text{CLASS}(r, d, a, Q, R) \in \text{FP}$ then $\text{QCMA} \subseteq \text{P}^{\text{NP}}$.*

Note that, $\text{QCMA} \subseteq \text{P}^{\text{NP}}$ would be very surprising because, among other things, it would imply $\text{BQP} \subseteq \text{PH}$, and there is evidence that this is false [1, 15].

Proof. The standard QCMA-complete language \mathcal{L} is as follows. Let \mathcal{C} be the set of all quantum circuits (expressed as a concatenation of bitstrings that index elements of the gate set Q). \mathcal{C} decomposes as the disjoint union of \mathcal{L} and $\bar{\mathcal{L}}$ where \mathcal{L} consists of the quantum circuits that accept

at least one classical (*i.e.* computational basis state) input, and $\bar{\mathcal{L}}$ consists of the circuits that reject all inputs. Given a quantum circuit $V_1 \in \mathcal{C}$, (the “verifier”) we can amplify it using standard techniques [29, 31] to accept YES instances with probability at least $1 - O(2^{-n})$ and accept NO instances with probability at most $O(2^{-n})$. Let V_2 be such an amplified verifier. Further, let



where the second-to-top qubit is the acceptance qubit of V_2 . If $V_i \in \bar{\mathcal{L}}$ then $\|V_3 - \mathbb{1}\| = O(2^{-n})$. By assumption, there exists a polynomial time classical algorithm for solving $\text{CLASS}(r, d, a, Q, R)$. When presented with V_3 , this algorithm will produce a polynomial-size reversible circuit V_4 strongly equivalent to the identity. By querying an oracle for the problem of strong equivalence of reversible circuits, one can decide whether V_4 is equivalent to the circuit of no gates, and hence to the identity operation. If $V_1 \in \bar{\mathcal{L}}$, this oracle will accept. If $V_1 \in \mathcal{L}$ then the algorithm for problem 1 will answer NO or produce a circuit that this oracle rejects. As shown in [26], the problem of deciding strong equivalence of reversible circuits is contained in coNP . Thus, we can decide QCMA in P^{coNP} , which is equal to the more familiar complexity class P^{NP} . \square

6 Future work

6.1 Dictionary attacks

The partial-indistinguishability obfuscator described in the preceding sections deterministically maps input circuits to obfuscated circuits. This creates a potential weakness in the obfuscation. Suppose Alice wishes to run a computation C on Bob’s server but does not wish Bob to know what computation she is running. Thus, she sends the obfuscated circuit $\mathcal{O}(C)$ to Bob, who executes it, and returns the result. Alice may hardcode the input to the circuit, and append a one-time pad encryption to the output, so that Bob learns nothing of C , the input, or the output. However, if Bob knows that the circuits Alice is likely to want to execute are drawn from some small set S , then Bob can simply compute $\{\mathcal{O}(s) | s \in S\}$ and identify Alice’s computation by finding it in this list. Such attacks are sometimes called “dictionary” attacks after the practice of recovering passwords by feeding all words from a dictionary into the hash function and comparing against the hashed password.

Dictionary attacks may or may not be a serious threat to our obfuscation scheme, depending on the size of the set of likely circuits to be obfuscated. In cryptographic applications where dictionary attacks are a concern, the standard way to protect against them is to append random bits prior to encryption. (In the context of hashing passwords, this practice is called “salting”.) Such a strategy can be applied to our obfuscator, but some care must be taken in doing so. The most obvious strategy is to append a random circuit on the output ancillas prior to obfuscation. However, attackers can defeat this countermeasure by using the polynomial-time algorithms for computing left-greatest-common-divisors in the braid group [14]. However, prior to obfuscation, one may introduce extra dits, and apply random circuits before, after, and simultaneously with the computation, in a way so as not to disrupt it. The problem of optimizing the details of this procedure so as to maximize security and efficiency is left to future work.

6.2 Classical and quantum universality

It is of interest to consider other computationally universal representations of the braid group, which might provide more efficient translations from circuits to braids. One avenue for obtaining such representations is by finding other solutions to the Yang-Baxter equation, besides the operator R from Section 4. Our investigations so far prove that no permutation matrix solution of dimension up to 16×16 is a universal gate and suggest that no permutation matrix solution of dimension 25×25 is a universal gate. In the quantum case, it has been shown that no 4×4 unitary solution is universal [3].

More generally, one may look for other finitely-generated groups with computationally universal representations and efficiently computable normal forms. One potential candidate family are the mapping class groups $\text{MCG}(\Sigma_g)$ of unpunctured surfaces of genus g . These groups also have quantumly universal representations [4] and an efficiently solvable word problem [21]. It is not known if there are also classically universal permutation representations, or if there are efficiently computable normal forms.

6.3 Expanding the set of indistinguishability relations

By [26], achieving efficient indistinguishability obfuscation for the complete set of relations of a universal gate set is unlikely. However, it is possible that partial-indistinguishability obfuscation on R gates could be achieved with a larger set of relations than the braid relations. For example, the universal reversible gate described in section 4 has order 60. If we add the relations $\sigma_i^{60} = \mathbb{1}$ for $i = 1, 2, \dots, n-1$ to B_n , we obtain a “truncated” (but still infinite for large n [11]) factor of the braid group. If a normal form can still be computed in polynomial time for this group then one could construct an efficient obfuscator using the relations of this truncated group, which would be strictly stronger than our braid group obfuscator. This approach also provides motivation for finding a complete set of relations for the gate R .

Acknowledgments

We thank Anne Broadbent, Rainer Steinwandt, Scott Aaronson, Bill Fefferman, Leonard Schulman, and Yi-Kai Liu for helpful discussions. We also thank Mariano Suárez-Alvarez and Gjergji Zaimi for leading us to reference [11] via `math.stackexchange` and `mathoverflow`.

References

- [1] Scott Aaronson. BQP and the polynomial hierarchy. In *STOC '10: Proceedings of the 42nd ACM symposium on Theory of Computing*, pages 141–150, 2010. [arXiv:0910.4698](#).
- [2] Dorit Aharonov and Itai Arad. The BQP-hardness of approximating the Jones polynomial. *New Journal of Physics*, 13(3):035019, 2011.
- [3] G. Alagic, S. Jordan, and A. Bapat. Non-universality of low-dimensional solutions to the Yang-Baxter equation. In preparation.

- [4] Gorjan Alagic, Stephen P. Jordan, Robert Koenig, and Ben W. Reichardt. Approximating Turaev-Viro 3-manifold invariants is universal for quantum computation. *Physical Review A*, 82:040302(R), 2010. [arXiv:1003.0923](#).
- [5] Emil Artin. Theorie der zöpfe. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 4:42–72, 1925.
- [6] B. Barak. Can we obfuscate programs? http://www.cs.princeton.edu/~boaz/Papers/obf_informal.html.
- [7] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO 2001*, number 2139 in Lecture Notes in Computer Science, pages 1–18. Springer-Verlag, 2001.
- [8] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973.
- [9] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.
- [10] Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation – tools for software protection. *IEEE Transactions on Software Engineering*, 28(8):735–746, 2002.
- [11] H. S. M. Coxeter. Factor groups of the braid group. In *Proceedings of the 4th Canadian Mathematical Congress*, pages 95–122, 1959. See <http://mathoverflow.net/questions/48849/>.
- [12] Christopher M. Dawson and Michael A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Information and Computation*, 6(1):81–95, 2006. [arXiv:quant-ph/0505030](#).
- [13] Patrick Dehornoy. Efficient solutions to the braid isotopy problem. *Discrete Applied Mathematics*, 156:3094–3112, 2008. [arxiv:math/0703666](#).
- [14] D. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson, and W. Thurston. *Word processing in groups*. Jones and Bartlett Publ., 1992.
- [15] Bill Fefferman and Chris Umans. Pseudorandom generators and the BQP vs. PH problem, 2010. [arXiv:1007.0305](#).
- [16] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4):219–253, 1982.
- [17] Michael H. Freedman, Michael Larsen, and Zhenghan Wang. A modular functor which is universal for quantum computation. *Communications in Mathematical Physics*, 227:605–622, 2002. [arXiv:quant-ph/0001108](#).
- [18] F.A. Garside. The braid group and other groups. *Quart. J. Math. Oxford Ser.*, 2, 20:235–254, 1969.

- [19] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *Theory of Cryptography - TCC 2007*, pages 194–213. Springer, 2007.
- [20] Juan González-Meneses. Basic results on braid groups, 2010. [arxiv:1010.0321 \[math\]](#).
- [21] Hessam Hamidi-Tehrani. On complexity of the word problem in braid groups and mapping class groups. *Topology and its Applications*, 105:237–259, 2000.
- [22] Jarmo Hietarinta. All solutions to the constant quantum Yang-Baxter equation in two dimensions. *Physics Letters A*, 165:245–251, 1992.
- [23] Edward V. Huntington. Sets of independent postulates for the algebra of logic. *Transactions of the American Mathematical Society*, 4:288–309, 1904.
- [24] Kazuo Iwama, Yahiko Kambayashi, and Shigeru Yamashita. Transformation rules for designing CNOT-based quantum circuits. In *DAC '02: Proceedings of the 39th annual Design Automation Conference*, pages 419–424, 2002.
- [25] Dominik Janzing, Pawel Wocjan, and Thomas Beth. “Identity Check” is QMA-complete, 2003. [arXiv:quant-ph/0305050](#).
- [26] Stephen Jordan. Strong equivalence of reversible circuits is coNP-complete. In preparation.
- [27] Louis H. Kauffman. *Knots and Physics*. World Scientific, 1991.
- [28] A. Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303:2–30, 2003. [arXiv:quant-ph/9707021](#).
- [29] Chris Marriott and John Watrous. Quantum Arthur-Merlin games. *Computational Complexity*, 14(2):122–152, 2005. [arXiv:cs/0506068](#).
- [30] Carlos Mochon. Anyons from nonsolvable finite groups are sufficient for universal quantum computation. *Physical Review A*, 67(2):022315, 2003. [arXiv:quant-ph/0206128](#).
- [31] Daniel Nagaj, Pawel Wocjan, and Yong Zhang. Fast amplification of QMA. *Quantum Information and Computation*, 9(11/12):1053–1068, 2009. [arXiv:quant-ph/0904.1549](#).
- [32] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2000.
- [33] R. Walter Ogburn and John Preskill. Topological quantum computation. In *Quantum Computing and Quantum Communications*, volume 1509 of *Lecture Notes in Computer Science*, pages 341–356. Springer, 1999. First NASA International Conference QCQC '98.
- [34] Peter W. Shor and Stephen P. Jordan. Estimating Jones polynomials is complete for one clean qubit. *Quantum Information and Computation*, 8(8/9):681–714, 2008. [arXiv:0707.2831](#).
- [35] Eric D. Simonaire. Sub-circuit selection and replacement algorithms modeled as term rewriting systems. Master’s thesis, Air Force Institute of Technology, 2008.
- [36] Simon Trebst, Matthias Troyer, Zhenghan Wang, and Andreas W. W. Ludwig. A short introduction to Fibonacci anyon models. *Progress in Theoretical Physics Supplement*, 176:384–407, 2008. [arXiv:0902.3275](#).